J M Bishop
20/3/90

South African Computer Journal
Number 1
January 1990

Suid-Afrikaanse Rekenaartydskrif
Nommer 1
Januarie 1990

Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels

The South African Computer Journal
An official publication of the South African Computer Society and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Suid-Afrikaanse Rekenaarvereniging en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

Subscriptions

	Annual	Single copy
Southern Africa:	R32-00	R8-00
Elsewhere:	$32-00	$8-00

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Editorial

At last the first edition of SACJ is available. I trust that readers will find it worth the waiting. There have been a number of teething problems in getting things together, the many details of which need not be spelt out here. One significant challenge was to cope with the consequences of the resignation of Quintin Gee, QI's highly competent production editor. He assisted in the initial phases of getting this publication together but had to resign for personal reasons. It is fitting to acknowledge here not only his initial advice and assistance in getting this first issue of SACJ off the ground, but also the many hours of work that he spent in previously producing QI.

Quintin's resignation meant that a new *modus operandi* for typesetting and printing had to be established. The exercise was not only time-consuming, but also has significant cost implications. Fortunately, the Unit for Software Engineering (USE) at Pretoria University has generously agreed to sponsor this first edition. On behalf of the South African computing community, I should like to thank them for their generosity. Now that they have made a first issue of SACJ possible, it is hoped to solicit the sponsorship of one of the larger computer companies for future editions.

It might be of interest to take readers on a walk through the new journal to highlight various aspects. To begin with, the cover design follows that of several journals whose titles have the format: *The South African Journal of Subject / Die Suid-Afrikaanse tydskrif vir Vakgebied* (where *Subject* and *Vakgebied* are appropriately instantiated). While colours vary, these journals generally have *Subject* and *Vakgebied* restated on the darker portion of the cover. SACJ's title was chosen in preference to a more descriptive but also more cumbersome title such as *The South African Journal of Computer Science and Information Systems*. The appearance of the words *Computer Science and Information Systems / Rekenaarwetenskap en Inligtingstelsels* on the cover are thus out of step with the original inspiration, but seem appropriate under the circumstances.

The inside cover is of interest for several reasons. Firstly, note that Peter Lay has kindly agreed to lighten my task by acting as an assistant editor. He will deal with matters relating to Information Systems. *Contributions in this area should henceforth please be sent directly to him.* Also note that an editorial board of distinguished persons has been assembled. I should like to once again thank board members for adding status to SACJ by agreeing to serve in this capacity. They will be consulted on matters of editorial policy whenever appropriate. Finally, the subscription costs have been increased to keep pace with production costs. This increase does not affect SAICS members, who will continue to receive the journal as one of the benefits of membership.

The guest editorial by Pieter Kritzinger makes for interesting reading. Several points of concern about computer-related research in South Africa are raised. I trust that the article will focus attention on these problems and stimulate a debate which will lead to eventual solutions. It is hoped to make guest editorials a regular feature of future SACJ issues.

Of the eight research papers offered in the journal, four have been gone through the normal channel of refereeing and revisions. The remainder were submitted to the Vth SA Computer Symposium and are published here by invitation. Each paper submitted to the chairman of the symposium's program committee was sent to three referees. A ranking scheme, reflecting an aggregate measure of referee evaluation, was used as a basis for deciding on papers to be presented. After further editorial evaluation, the authors of four of the five highest ranking papers were invited to submit their papers to SACJ. While it was not possible to contact the fifth author in time for this edition, but it may be possible to publish that paper, together with a selection of others from the symposium, in future SACJ editions.

In the section marked *Communications* various items of news arriving at the editor's desk have been published. It was particularly gratifying to receive book review submissions in response to a prior general appeal. There has also been an enthusiastic response from book publishers, who have sent in a number of books for review. Titles are listed in the *Communications* section. Please contact me if you are willing to review one (or more) of these. Naturally, reviews of other books of interest in your possession will also be welcomed.

The final point to highlight in this walk through the journal is the increase in page charges indicated on the back inside cover. These reflect the increased cost of production. Since research papers in SACJ qualify for state subsidy at academic institutions, the charges should not, in general, present major problems for authors. However, it is worth pointing out that the final format of papers submitted significantly impacts on both the financial and editorial load. Submissions in camera-ready format (or nearly so) result in both a cost savings and a speed up of turn-around time by several orders of magnitude. Since many readers may not be familiar with the printing process, it may be helpful to say something about it in order to substantiate this claim.

The printing process basically involves typesetting, shooting (or photographing), and then reproduction and binding. Apart from limiting the amount of material, the printer's client has very little control over the cost of shooting, reproduction and binding. On the other hand, anyone equipped with moderate text- or word processing facilities and a laser printer can go a long way (if not all the way) towards typesetting a paper. Even a partially typeset paper helps significantly, as I will explain below.

By typesetting I simply mean knocking the paper into the right shape and producing a laser printout. The printers regard this is a tedious, error-prone task, even if they start off with an ASCII file rather than a hardcopy of the paper. Consequently, they tend to handle large-scale typesetting by subcontracting the task. Moreover, while they may be willing to typeset uncomplicated text, they tend to balk at text containing specialized mathematical and other notation. However, they are quite skilful at cutting and pasting text, and at enlarging or reducing photographed or scanned diagrams. They are even willing to redraw sketches which are not too complicated.

As a result of the above, I have pressed several authors to do their own typesetting. In cases where it was problematic to produce double column format, a single column of appropriate width was requested. While this is a second-best option, it allows for cutting and pasting to be done by the printers. Some sketches have either been directly reduced from the author's original, while others have been redrawn by the printer. By way of exception, I have personally undertaken the typesetting of a few papers using WordPerfect. However, I would like to avoid this as far as possible in future, and consequently appeal to potential authors to make every effort to do their own typesetting.

From SACJ's point of view encouraging authors to do their own typesetting involves a compromise in that there will inevitably be slight variations in the print from one article to the next (as is in fact the case in this issue). If you are pedantically inclined, you might consider this to be a disaster. Personally, I regard it as a rather neat advertisement for the typesetting skills of SACJ contributors.

As an aside, since the handling of $\TeX$ files was initially a problem for me, I was pleased to discover that Peter Wood and his colleagues at UCT have mastered the art of producing $\TeX$ printout in the format now before you. Future authors who use $\TeX$ should consult them on details.

As to the future, it is not possible at the this stage to commit to a fixed number of SACJ issues per year. The number of issues is constrained by finance, submissions of the right quality, and time available to the editorial staff (including our anonymous and unsung heroes - the referees). The ideal is to produce four issues per year, but this may not always be attainable.

In conclusion, if readers have as much fun in reading this first issue of SACJ as I have had in editing it, the hours spent on it will have been well worthwhile. Hopefully SACJ is destined not only to be a permanent feature of the Southern African computing scene, but also to significantly contribute to research in the region.

Derrick Kourie
Editor

This SACJ issue is sponsored by
The Unit for Software Engineering
(USE)
Department of Computer Science
Pretoria University

Funding Computer Science Research in South Africa

P S Kritzinger
Department of Computer Science, University of Cape Town, Rondebosch 7700

The word *research* has many connotations and is often abused. In everyday language a person does not simply *search for information in a library*, for example, but rather does *research*, thus pretentiously conferring an aura of intellectual activity on an effort which requires very little original thought.

Here I will interpret the term to mean work which generates results that gain international recognition. This implies that the work is published in good international journals or presented at international conferences. I believe this is the only valid index of the quality of research.

With very few exceptions, the computer industry in South Africa is a consumer of computer technology, rather than a developer. In contrast with, say, the chemical industry, there is therefore no tradition of research in computer science in the South African computer industry and computer science researchers therefore have, as virtually their only source of funding, the Foundation for Research Development (FRD) which has its origins in the CSIR.

The FRD was formed in April 1984 with the development and use of research expertise in the natural and applied sciences and engineering as its mission. This mission is primarily directed at the universities, museums and technicons with the ultimate aim of improving the life of all South Africans.

Although the FRD has several programmes, the two which are of main concern to computer scientists are the Core Programmes and the Special Programmes.

FRD Core Programmes foster the optimum development of a scientific and technological knowledge base by supporting individual self-initiated research. These programmes, started only about 4 years ago, have met with considerable acclaim, particularly in regard to the way in which research funding for a particular individual is decided. To qualify for support within a Core Programme, researchers must obtain a certain evaluation status within the FRD and funding is then linked directly and exponentially to the merit of the individual concerned, rather than being linked to the specific project proposed.

In the evaluation process, peer review is strongly emphasised. The researcher himself is expected to nominate referees, whose status and reports play a decisive role in the evaluation. As a result of this evaluation, an applicant is assigned a specific evaluation status category. There are currently 9 categories in all, but the ones of main interest are:

A researchers who are without any doubt accepted by the international community as being amongst the leaders in their field (52);
B researchers not in category A but who nevertheless enjoy considerable international recognition as independent researchers of high quality (182);
C proven researchers who have maintained a constant high level of research productivity and whose work is regularly made known internationally, or proven researchers whose current research output is less but who are actively engaged in scholastic activity (433);
P researchers younger than 35 years of age who have already obtained a doctoral degree and who have shown exceptional potential as researchers (10); and
Y young researchers usually under 35 years of age, who are highly likely to achieve C status by the end of their support period (108).

The number of researchers in the various categories as of August 1989 has been indicated in parentheses above. Of these, only 7 persons are computer scientists: 1 in category B; 3 in category C; and 3 in category Y. Only 4 departments of computer science are involved.

The other main programmes of concern to computer persons are the Special Programmes which aim at developing research manpower in priority areas. After identification of an area that merits particular research development, given local expertise, a Special Programme is launched to address the problem in the national interest.

Although a manager of a Special Programme has to be an FRD evaluated researcher, the same need not be true for the other team members. Regular peer evaluation of researchers as well as evaluation of the progress and results of Special Programmes are considered essential. Special Programme awards will be made for the first time towards the end of 1989. It is therefore not yet known whether proposals already submitted for programmes in computer science have been successful.

It is clear that, in the context explained above, there is virtually no computer science research being done in South Africa - a scary thought which has considerable implications for this country! Why is this so? There are several reasons, but I would like to single out two in particular.

Qualified faculty and students is an abiding problem at the heart of computer science departments. Acquisition of new faculty members is an issue intimately linked to the number of graduate students successfully completing PhD degrees. This problem is by no means unique to South Africa. For instance, data gathered in North America indicates that in 1983 there were over 200 vacancies in the 91 departments that have doctoral programmes in computer science. At the same time, only approximately 250 PhD's were granted in North America - a figure that has remained relatively unchanged for the past several years. A large number of those graduates were attracted to industry and industrial research laboratories. Although I do not have solid data at my disposal, I would think that South Africa produces at most one PhD graduate in computer science per year. There are currently 20 departments of computer science at universities is South Africa. It will therefore take us 20 years to locally produce one new faculty member with a PhD in computer science for every university.

Contributing to the above problem is our current academic image. The graduate student usually sees concerned computer science faculty members as rather harried individuals, having large undergraduate classes, much committee and professional work, and labouring under an ill-fitting model (applicable to more established disciplines) for decisions on tenure, salary and promotion. Further, as undergraduates, many prospective graduate students were not engaged in research projects involving computer science faculty, and for that reason were not exposed to graduate students doing research, and rarely developed a camaraderie with any computer science professionals. At last count there were only 5 individuals in South Africa who completed their computer science doctorate at a university outside South Africa where they had the good fortune to work in an environment in which sufficient faculty and funds were available to create an ethos of research. It is difficult to convince students that their interests and goals can be served by a PhD in computer science or by an academic career.

The second problem, which is of greater concern to me since there is no immediate solution to it, has to do with the fact that senior persons who decide the fate and fortune of academic computer science departments are, in general, individuals whose professional careers started well before computing machines came into every day use - that is to say, in the years B.C. (Before Computers). These persons of influence do not always understand what "computers" are, and what their potential influence upon the workplace in particular and society in general are. As far as research (as opposed to teaching) is concerned, most of them understand that a medical school needs special and expensive equipment (not to mention, expensive faculty) and that engineers must have a workshop and special machinery to teach their students and conduct research. They understand that if one needs to build up a defense industry, it will cost billions of rands; but they are not so sure about computer science, even though many other countries have recognised it as of national strategic importance.

I believe that only time and dedication will lead to a solution of these seemingly insurmountable problems and allow computer scientists to take their rightful place in the research community in South Africa.

Bibliography

P J Denning, [1981], Eating our seed corn, *Communications of the A.C.M.*, **24**(6), pp.341 - 343.

J Tartar (Ed.), [1985], The 1984 Snowbird Report: Future issues in computer science, *Communications of the ACM*, **28**(5), pp490 - 493.

D Gries, R Miller, R Ritchie and P Young, [1986], Imbalance between growth and funding in academic computing science: two trends colliding, *Communications of the A.C.M.*, **29**(9), pp.870 - 878.

J E Hopcroft and D B Krafft, [1987], Towards better computer science, *IEEE Spectrum*, pp.58 - 60.

The NRDNIX Distributed Database Management System

M.H. Rennhackkamp
Department of Computer Science, University of Stellenbosch

Abstract

A distributed database management system must facilitate integrated processing of data which is physically distributed over a network of node computers, while providing locality transparency.

The NRDNIX prototype has locality, logical and physical independence as goals, together with adequate availability and efficient throughput. It is architecturally described in terms of an integration of the ANSI/X3/SPARC database management system and the ISO interconnection reference models. Communication takes place via a broadcast network. The data is horizontally distributed over the horizontally controlled nodes, with duplication according to usage. An adaption of the entity-relationship data model is supported, with extensions to the relationships to include update dependencies.

Some supporting operating systems have been considered. The most promising approach seemed to be the embedding of the distributed database management system kernel in the MINIX operating system. This approach accentuated shortcomings of the MINIX operating system, which deemed it inappropriate for the implementation. A more feasible and popular implementation currently being used is the development of device drivers for the XENIX operating system, for example for the network interface.

Although the NRDNIX prototype is nearing completion, many related research projects must still be undertaken to complete it into a product.

Keywords: *Database management, Distributed database system*
Computing Review Category: *H.2*

Received August 1988, Accepted April 1989

Introduction

This paper assumes that the reader has a general working knowledge of databases, networks and distributed databases. These are adequately documented in introductory texts by Date [7], Ceri and Pelagatti [2] and Tannenbaum [12].

The goal of the NRDNIX project is to implement a homogeneous distributed database management system on a high speed network of powerful personal computers (eg PC ATs). The aim is to provide horizontal control, concurrent distributed operations and locality independence. The entity-relationship data model is supported. The implementation of this data model, as well as the interface between the distributed database management system and the operating system has already led to some interesting results.

The first section overviews the architecture of the distributed database management system, in the context of standard database management system and network architectures. This establishes a distributed database management system reference model. The second section very briefly overviews the major design decisions taken. The third section discusses the implementation of the entity-relationship model. The fourth section addresses the interface between the supporting operating system and the distributed database management system, noting experience gained so far with the embedding of the communication kernel in the MINIX operating system. The fifth section gives a status report of the development of the various modules. The paper is concluded with an overview of proposed extensions.

The project is undertaken by the Study Group on Distributed Databases NRD/01(86) of the Department of Computer Science at the University of Stellenbosch, from which its name is derived.

1. DDBMS Reference Model

The distributed database management system (DDBMS) reference model is established by the integration of the ANSI/X3/SPARC 3-levelled centralized database management system (DBMS) architecture [14] and the ISO 7-layered network model for open system interconnection (OSI) [12]. It can be illustrated by the diagram of figure 1.

On the conceptual level, the integrity control of the DBMS and the network session and transport layers are explicitly integrated in the communication kernel. The communication kernel is the central control centre of the architecture at each node.

On the external level the network presentation module and the DBMS external model are coalesced as the user interfaces in the presentation manager.

The database manager controls the DBMS read and write primitives at the internal level as part of the accessing method of the DBMS, independent of the network architecture.

The network manager (eg the subnet control functions of the network, data link and physical layers) functions independently of the DBMS integration, to provide an error free communication service to the communication kernel.

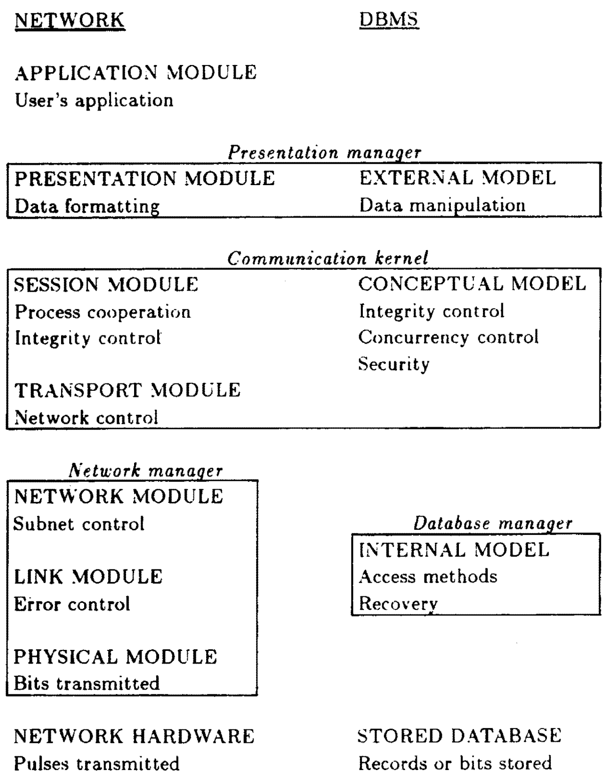

Figure 1: DDBMS architectural reference model

The functional flow between these identified components at a single node are shown in figure 2.

Two important components not shown on the diagram are the data dictionary and the recovery manager. The data dictionary must store the metadata and provide it to the presentation manager, communication kernel and database manager modules. The recovery manager must provide for the keeping of redundant recovery data during normal processing and it must control the recovery process when a failure occurs.

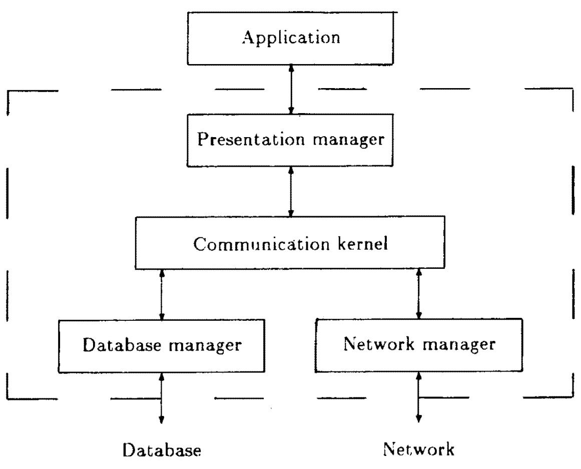

Figure 2: Functional flows

2. Design Aspects

In this section the various chosen design options are discussed.

2.1 Component DBMS

The component DBMSs of the NRDNIX DDBMS are homogeneous. Each component supports the same data model and access language facilities. Although the structure of the nodes is similar, they are not limited in number; thus allowing dynamic expansion as long as all the nodes conform to the homogeneity.

2.2 Control Distribution

The control between nodes in a DDBMS include message traffic management, update synchronization and recovery.

A horizontal control distribution was chosen for improved reliability and expandability. All processes cooperate at the same logical level. The data and control flow is managed by the node establishing the connection. Some aspects (eg two-phase commitment) need a controlling function, but this is usually only desired temporarily and can be achieved by dynamic master-slave switching, where the node initiating an action performs the duties of the master for the duration of the action.

The advantage of this configuration is that a failure of the single central controlling node cannot affect the whole DDBMS.

2.3 Data Representation

Duplication occurs where active copies of a specific data item are stored at various nodes. All or part of the conceptual DDB is replicated at two or more nodes.

Daniell et al [6] presented a replication (duplication) technique which takes advantage of the following characteristics:

- Geographic affinity: Accesses to a given data item tend to cluster geographically, meaning that data items accessed at a given node are more likely to be accessed again at that node.
- Temporal affinity: Accesses to a given data item tend to cluster in time, meaning that data items that have been recently accessed are more likely to be accessed in the near future.

The node at which the accesses for given data item tend to cluster is called the affinity node of the data item; the affinity node need not be known in advance and it may vary with time. The result is that data items are dynamically copied as required and dynamically reduced at particular sites if not actively used.

The redundancy induced by duplication improves read-only accessibility through reduced internode communications. It consumes additional storage, increases update communication costs and incurs more complex consistency and concurrency control.

2.4 Data Dictionary

The data dictionary is the central repository and inventory of the data resource. It contains metadata (data about the database data).

With only a local data dictionary at each node, no additional information is kept apart from the data structures locally available. This approach might induce high communication costs in a vertically partitioned environment; but with dynamic data duplication being used, the entire data model structure is available from the data dictionary of each site.

2.5 Accessing Method

The access method determines where processes responsible for the executions on behalf of transactions are located and executed if requests cannot be solved locally.

Transaction switching is where a transaction is moved to the remote data and is processed there, returning the desired results. The problem of remote data duplicated at various nodes, is solved by split processing.

Split processing occurs when a transaction is divided into subcomponents which are then executed at remote sites. When concurrent splitting is used, relevant parts of the request are sent to all the appropriate nodes, where these are processed concurrently and from where the results are then merged at the originating node.

2.6 Concurrency Control

Concurrency control entails the method used to ensure that multiple concurrent updates are processed in a serializable manner.

The method used in NRDNIX was chosen according to a comparative study of popular concurrency control methods, based on overhead costs as an evaluation criterion [11]. Conservative time-stamp ordering has the least overheads, for horizontal control in a homogeneous configuration with dynamic data replication, with due consideration to communication costs. It is utilized with improvements on the buffering techniques for improved throughput on files with few conflicts.

3. Entity-relationship Data Model

Chiang and Bergeron [5] described a DBMS implementation based on the three levelled ANSI/X3/SPARC architecture, with the entity-relationship data model used for the conceptual level. It was slightly modified from the original entity-relationship data model as described by Chen [3], to capture more semantics in the form of update dependencies.

3.1 Data Structure

The data structure consists primarily of entities and relationships, each with optional descriptive attributes.

3.1.1 Entity

Entities are objects or concepts, which can be identified and which have independent existence.

A set of entity occurrences can be represented by a relation, as a subset of the cartesian product over the collection of domains of the relation. Each domain represents a value set. The semantics of entities possessing certain properties are captured by attributes. An entity attribute is a function mapping from the entity set into a value set of a domain of the relation. A set of attributes, which is a one-to-one function, is designated as the primary key of the relation.

Thus, an entity is conceptually defined by the following:

- Name.
- Attributes.
- Key attributes.

3.1.2 Relationship

A relationship represents an association between entities. Although it can be depicted by a relation with at least two domains of entity set attributes, it is semantically different from a relation. The definition of an association among entities is enhanced with a type of mapping and a coupling factor to capture more semantics of update dependencies.

The type of mapping, which conveys the semantic information about the number of entity occurrences involved, can be one-to-one, one-to-many or many-to-many.

The coupling-factor, which captures the closeness of the two entity classes, can be very tight, tight, medium tight or loose. For example in a relationship A with entity domains E_1 and E_2, the following holds.

A has a very tight coupling-factor from E_1 to E_2 iff:

a) An insertion of entity e_2 in E_2 is permitted only if the associated e_1s are in E_1;
b) A deletion of e_1 from E_1 implies the deletion of all e_2 in E_2, for all (e_1,e_2) in A; and
c) No deletion of e_2 without deletion of e_1 is permitted for any (e_1,e_2) in A.

A has a tight coupling-factor from E_1 to E_2 iff A satisfies conditions a) and b) above.

A has a medium tight coupling-factor from E_1 to E_2 if deletion of e_1 from E_1 is permitted iff no e_2 in E_2 is associated with e_1 via A; insertion of e_1 is independent of e_2.

A has a loose coupling-factor iff for e_1 in E_1 and e_2 in E_2, where (e_1, e_2) is in A, e_1 and e_2 can be inserted and deleted independently.

Thus, a relationship is conceptually defined by the following:

- Name;
- Entities participating in the relationship;
- Type of mapping between the entities;
- Type of coupling between the entities; and
- The attributes of the relationship.

3.1.3 Attribute

Attributes are the named characteristics of entities or relationships. An attribute is defined as being of a specific type, with elements taken from a specified domain of possible values. Often the logical size of an attribute is also specified, but this is implementation dependent and does not have semantic meaning.

An attribute is conceptually defined by the following:

- Name;
- Type;
- Underlying domain or range; and
- Size (logical).

3.2 Sublanguages

The external and conceptual level sublanguages are the data definition language (DDL) and the data manipulation language (DML).

3.2.1 Data Definition Language

The DDL (facility) is menu and screen oriented. The number of functions to be performed are few and well defined; it does not warrant using a syntactical DDL with complicated entry, parsing and interpretation.

3.2.2 Data Manipulation Language

The DML is non-procedural and non-navigational in nature; it specifies what must be done and not how. The results of DML statements are tabular representations of entities and relationships with their attributes. However, the DML statements are syntactically specified, to provide the users with a powerful retrieval capacity and to facilitate retrieval commands embedded in host programming languages.

The retrieval command of the NRDNIX implementation is based on the non-procedural form of the SQL SELECT statement:

SELECT attributes
FROM {relations / entities}
WHERE condition

The insertion command is based on the tuple or (attribute; value) pair notation:

INSERT
{<*value, value,...,value*> (tuple notation) /
(*attribute, value*), (*attribute value*),...(pair notation)}
IN {entity / relationship}

The deletion operation can only be specified for a single entity or relationship, but it may be based on a complex selection condition:

DELETE
FROM {entity / relationship}
WHERE condition

The update operation can also only be specified for a single entity or relationship, but it may be based on a complex condition:

UPDATE attributes
WITH values
IN {entity / relationship}
WHERE condition

The DML will later be adapted to comply with the standard SQL syntax, with the addition of the optional ENTITY and RELATIONSHIP keywords.

4. Operating System Interface

Due to the fact that data duplication and transaction splitting is utilized, the underlying operating system (O/S) must offer the facility to run concurrent processes.

4.1 Alternatives

The only commercially available O/S which complies with the above requirement is UNIX, or one of its look-alikes such as XENIX. These are the only O/S's which support concurrency control, for the NRDNIX hardware of high-powered small processors, such as PC ATs. The alternative is a custom tailored O/S or an academic experimental operating system such as Edison [1]. At this stage this involves too much in the form of learning, adaption and complexity combined to be practical.

The O/S initially chosen for the implementation is MINIX, which was designed and developed by Tanenbaum [13]. MINIX provides a substantial subset of the UNIX functions, with a usable development environment. Like the UNIX O/S, it was written in C; but it was designed in a much more structured layered architecture, based on the client-server message passing model of process synchronization and inter-process communication. In addition, the O/S source code is available, which facilitates the embedding of the lower level database manager and network manager modules very efficiently.

4.2 Problem Areas

However, during the research and initial stages of the embedding process, a number of serious problems were encountered.

The code of the MINIX O/S was designed in a structured manner and documented as such in the accompanying text [13]. However, in the actual MINIX implementation some modules have been changed to function more efficiently; these changes then breach the model of message passing which was described in the text.

Some examples have also been found in the MINIX O/S where the implementation of the message passing model does not function correctly in a multiprocessing mode. Deadlock can occur between a device driver (server) and the file system (client), because in allowing the device driver to notify the file system when an event occurs (eg an expected character is received from a terminal), the roles of the master (client) and slave (server) are reversed. Deadlock can occur when the real master and the temporary master wait for each other to rendezvous with messages to be delivered.

There are also some inherent shortcomings of the MINIX O/S. For example, a single global alarm is used to activate all waiting processes. However, if any process waiting on the alarm is killed, the alarm is also deactivated, which in turns causes all waiting processes to be delayed indefinitely or until another active process activates the alarm.

The difference of the lower level structures of the MINIX O/S, when compared to UNIX, would have adversely affected the DDBMS's portability. Some of the low level aspects had to be reprogrammed as device drivers in the UNIX O/S if a porting was to take place to a set of larger UNIX machines.

Device drivers are currently being developed, as the entire development has been transferred to the XENIX operating system. XENIX is a more commercially accepted operating system and does provide for greater portability.

These MINIX problems are discussed elsewhere in detail [9].

5. Status Report

A prototype skeleton of the NRDNIX DDBMS is nearing completion, which would demonstrate the commercial feasibility of the project, as it addresses the important critical aspects.

5.1 Data Dictionary

The distributed data dictionary already exists as an independent system module. At this stage the data dictionary merely maintains a number of files, corresponding to the system relations, with the necessary metadata manipulation operations. In the second version of the project the data dictionary management will be done by the database manager and the system relations containing metadata will be represented as standard entity and relationship relations.

The data dictionary is currently being used by the query decomposition method and user interface components of the presentation module. It is also utilized by the communication kernel to obtain the names of physical files used to represent logical entities and relationships. It is also used by the database manager to maintain the physical details of stored relations.

5.2 Presentation Module

The presentation module currently has simple DDL and DML interfaces. These can be used to define structures, insert data, manipulate data and execute queries on the data.

The global SQL-like queries are parsed and checked for correct syntax, including the correct use of data elements defined in the data dictionary. During parsing, a query operator tree is constructed; the operators contained in the tree are simple relational algebra operations, eg unary select, unary project, binary join and binary union operations. The operations contained in the query operator tree are then optimized using the Apers, Hevner and Yao algorithm documented in Ceri and Pelagatti [2].

The optimal sequence of relational algebra operations are then passed to the communication kernel, which controls their broadcasting and execution.

The presentation module calls the recovery manager to log global transactions.

5.3 Communication Kernel

The communication kernel contains the conservative time-stamp ordering based concurrency control module for the correct scheduling of the concurrent split transaction operations. The operations are buffered in time-stamp ordered lists per file. The communication kernel obtains the physical file names from the corresponding entity and relationship names used in the operations by consulting the data dictionary module.

During the execution of the concurrent subtransactions, the communication kernel also calls the recovery manager to write the logical logs of subtransactions. In the event of a failure, when a site returns to normal processing, it activates the recovery manager to perform the necessary recovery actions in the form of undo and redo operations. The recovery manager decides to undo or redo uncompleted transactions by consulting the log; a decision is made by comparing the current time-stamp with the logged transaction time-stamps.

5.4 Network Manager

The network manager provides a reliable error-free communication service to the communication kernel; it can be used in a broadcast or point-to-point mode. It can also be used by other programs and facilities.

The network system used is ArcNet, which provides a virtual token ring protocol on a bus architecture.

The network facilities were developed using a XENIX device driver. The device driver is an interrupt handler which buffers incoming messages and then it indicates to the network manager that a message has been received. The network manager is the process which controls the communication protocol, making use of the device driver [10].

5.5 Database Manager

The database manager is currently being developed. It consists of two major components, namely a cache manager and an access manager. The cache manager is responsible for maintaining working sets of data records in memory, in order to execute the low level data manipulation operations efficiently. The access manager is responsible for managing data records in files, using efficient access methods such as dynamic B-tree indexes. It forms the interface between the cache manager and the XENIX file system.

Conclusion

The first phase of the NRDNIX DDBMS is scheduled for completion at the end of 1989. The result will be a working prototype DDBMS. The project then enters a second phase, concerned with improvements on the first phase. Aspects which are being considered include alternative representations of the entity-relationship data model with a full SQL-like interface, alternative recovery facilities and improved forms of concurrency control such as time-stamped versions. Numerous other facilities, each representing another related field of study, can also be included in the system. Examples include security facilities, DBA interfaces and facilities, performance measurement and improvement facilities, as well as a 4GL interface.

This implies that there are still a number of available honours and M.Sc study topics, as well as implementation projects, under the NRDNIX banner which need attention.

References

[1] P Brinch Hansen, *Programming a Personal Computer,* Prentice-Hall, 1982.

[2] S Ceri & G Pelagatti, *Distributed Databases - Principles and Systems,* McGraw-Hill, 1985.

[3] P P Chen, The Entity-Relationship Model - Towards a Unified View of Data, *ACM Transactions on Database Systems*, 1(1), 1976.

[4] P P Chen (ed), *Entity-Relationship Approach to System Analysis and Design,* North-Holland, 1980.

[5] T C Chiang & R F Bergeron, A Data Base Management System with an Entity-Relationship Conceptual Model, In [4].

[6] T P Daniell, R C Harding jr & S H Nauckhoff, Dynamic replication: an overview, *AFIPS National Computer Conference,* AFIPS Press, 1983.

[7] C J Date, *An Introduction to Database Systems*, 4th ed, Addison-Wesley, 1986.

[8] G Gardarin, A Unified Architecture for Data and Message Management, *AFIPS National Computer Conference,* AFIPS Press, 1979.

[9] M D Meumann & M H Rennhackkamp, MINIX for a Distributed Database Management System, To appear, 1989.

[10] M D Meumann, Implementing the network levels in XENIX, *Technical Report,* Department Computer Science, University of Stellenbosch, 1989.

[11] M H Rennhackkamp, Comparison of Concurrency Control Methods in Distributed Databases, *M.Sc thesis,* Department Computer Science, University of Stellenbosch, 1986.

[12] Tannenbaum A.S., Computer Networks, Prentice-Hall, 1981.

[13] A S Tannenbaum, *Operating Systems: Design and Implementation,* Prentice-Hall, 1987.

[14] D Tsichritzis & A Klug (eds), The ANSI/X3/SPARC DBMS Framework: Report of the Study Group on Data Base Management Systems, *Information Systems,* 3(3), 1978.

Acknowledgement

The Institute of Applied Computer Science (ITR) of the University of Stellenbosch provided the hardware for the research project.

Computers and the Law

Submitted by Antony Cooper
CSIR

The SA Law Commission has established a commission on "The Legal Protection of Information".

The commission is still in its preliminary stages and the assigned researcher, Mr Herman Smuts, is still preparing the working paper. He does not know when it will be finished, but once the working paper has been prepared, they will invite comments for about two years, before preparing the final report. I have contacted Mr Smuts, and he would be most grateful to receive input at this stage, especially regarding the terms of reference of the commission. His address is:

C/o SA Law Commission
Private Bag X668
PRETORIA
0001

In addition, there is an ad-hoc committee at the Registrar of Copyright investigating numerous copyright issues, including those relating to software and data. Mr Smuts' commission will be liaising with the ad-hoc committee.

I feel that SAICS has an obligation to submit evidence to the commission, and I would appreciate it if you would circulate the members of the Council of SAICS, and perhaps the general membership as well, to solicit ideas concerning SAICS's input.

I shall prepare something for the commission, either in my personal capacity, or in my professional capacity here at CSIR. I would be willing to assist in the preparation of any evidence SAICS might submit.

4th National MSc/Phd Computer Science Conference

Report by Danie Behr
University of Pretoria

This conference was held from 7th to 10th September 1989 at the Cathedral Peak Hotel in the Drakensberg. The conference was attended by 61 postgraduate students from 11 South African universities. Most were engaged in MSc studies, although 5 Phd students also attended. These numbers are encouraging for the South African computer science community. This type of conference is rather unique in that it affords students the opportunity of sharing their research, and getting to know other researchers in the country. The number of Afrikaans and English speaking students attending the conference were roughly equal. Presentations were made in the language preferred by the student. Invitations were sent to all universities with computer science departments. The conference was organized by the students themselves.

Some of the more popular research topics that were presented included expert systems, data communications, computer security, graphics, software engineering, user interfaces and data bases. The main sponsor for this year's conference was the Division for Microelectronic Systems and Communication Technology of the CSIR. The conference was opened with an interesting talk on the myths and motivations of post graduate studies by Prof DG Kourie, acting head of the Computer Science Department at Pretoria University.

The next conference will be presented by the University of Port Elizabeth. People requiring further information about the next conference should contact Andre Calitz, Charmaine du Plessis or Jean Greyling of the Department of Computer Science at UPE.

A list of authors and papers presented at the symposium follows:

S Crosby, University of Stellenbosch
Performance Analysis of Wide Area Computer Communication Networks

A B Joubert, PU for CHE Vaal Triangle Campus
Image Processing Libraries

A Calitz, University of Port Elizabeth
An Expert System Toolbox to assist in the classification of objects

L von Backström, University of Pretoria
Integrated Network Management

R Foss, Rhodes University
The Rhodes Computer Music Network

A McGee, University of Natal
On Fixpoints and Nondeterminism in the Sigma-Lambda Calculus

P G Mulder, Randse Afrikaanse Universiteit
A Formal Language and Automota approach to Data Communications

A Tew, Randse Afrikaanse Universiteit
Drie dimensionele grafiek grammatikas

T C Parker-Nance, University of Port Elizabeth
Human-Computer Interaction: What Determines Computer Acceptance

E Coetzee, PU vir CHO Vaaldriehoekkampus
Opsporing van rande in syferbeelde dmv verskerping en drempelbepaling
D A Sewry, Rhodes University
Visual Programming
A Cooper, University of Pretoria
Improvements to the National Exchange Standard
E S Badier, University of Port Elizabeth
A Computer Assisted Diagnostic System (CADS)
C du Plessis, Universiteit van Port Elizabeth
Persoonsidentifikasie dmv naampassing in 'n genealogiese databasis
J Greeff, University of Stellenbosch
The Entity-Relationship Model and its Implementation
D A de Waal, PU vir CHO
Flat Concurrent Prolog (FCP) en Flat Guarded Horn Clauses (FGHC): 'n Vergelyking
E Naude, UNISA
Interne metodes in Liniêre Programmering
A Deacon, University of Stellenbosch
Global consistency in non-locking DDBMS
A Wilks, Rhodes University
The Synchronisation and Remote Configuration of the Resources in a Computer Music Network
J Greyling, University of Port Elizabeth
The design of a User Interface with special reference to an Interactive Molecular Modelling Program
L Drevin, PU vir CHO
Rekenaarsekuriteit: Verskillende vlakke van kontrole
Dieter C Barnard, University of Stellenbosch
The design and implementation of a modest, interactive proof checker
R A Schmidt, University of Cape Town
Knowledge Representation Systems and the Algebra of Relations
J Hartman, Randse Afrikaanse Universiteit
Die Gebruik van Objek-georiënteerde Programmering in die Moderne Sneltrein Omgewing
S Lawrie, Rhodes University
The Design and Implementation of a System for the Interactive Control of a MIDI-based Studio
E Mulder, Rand Afrikaans University
A Formalisation of Object-Oriented Principles
C J Tolmie, UOVS
Die Ontwikkeling van 'n Ekspertrekenaarstelsel vir die beoordeling van die resultate van die Technicon H1-Bloedselanaliseerder
R Breedt, University of Pretoria
Realism with Ray Tracing
J van Jaarsveld, University of Pretoria
Developing Medical Expert Systems: A knowledge acquisition perspective
W Appel, University of Pretoria
TCP/IP Implementation on Ethernet
E Goedeke, University of Natal
Eggspert's Control Structure
M Harmse, University of Stellenbosch
Modelling of I/O Subsystems
H L Viktor, University of Stellenbosch
A Quantitative Model for Comparing Recovery Techniques in a Distributed Database
M Olivier, Randse Afrikaanse Universiteit
Rekenaarvirusse in Suid-Afrika

Book Reviews

An Introduction to Functional Programming Through Lambda Calculus
by Greg Michaelson, Addison-Wesley, 1988.
Reviewer: Dr. E P Wentworth, Rhodes University

Recently we have seen a number of excellent *second generation* texts on Functional Programming. Michaelson's text assumes some previous programming experience with imperative languages, and presents the functional approach as an alternative paradigm. He begins with a very accessible exposition of the Lambda Calculus, and carefully develops this foundation to encompass the important aspects and paradigms of functional programming. The programming notation is language-independent, although the last chapters are devoted to a brief look at two specific languages, Standard ML and Lisp. The examples and exercises are mainly utility in nature, e.g. "insert a sublist after the first occurrence of another sublist in a list", and can generally be solved in a couple of lines. Answers to the exercises are provided in an appendix.

The approach is slanted towards developing a solid base for understanding functional languages and computing. In this respect the book achieves a good balance between the theoretical underpinnings and their practical application. On the practical side, however, I found the lack of more substantial examples and exercises disappointing. Most programming texts tackle a set of 'standard' problems which are well-understood in the academic community and provide an informal benchmark for comparisons. Since the book is targeted for those already versed in imperative languages and standard algorithms, one might expect the examples to clearly demonstrate the elegance and power of the *problem-oriented* functional approach in these areas. Having laid an excellent foundation I was left with the feeling that the book failed to capitalize and deliver the cherry on the top.

The book is highly recommended as one of the new breed of Computer Science books which gives substantial attention to the fundamentals of the subject without becoming bogged down in over-rigorous formality.

Artificial Intelligence and the Design of Expert Systems
by George F Luger & William A Stubblefield, The Benjamin/Cummings Publishing Co., 1989.
Artificial Intelligence: A Knowledge-based Approach
by Morris W Firebaugh, PWS-Kent Publishing Co., 1989.
Reviewer: Prof G D Oosthuizen, University of Pretoria

One of the primary goals of an Honours course is to introduce students to a field in such a way that they arrive at enough insight into relevant issues to enable them to conduct further research on their own. To this end a text book which is used ought to reflect the current view of the field. Because of the rapid expansion of the field of Artificial Intelligence (AI), we have now finally outgrown the era dominated by the books by Winston and Charniak and McDermott. In the past five to ten years much new work has been done, and new insights have been gained. Introducing AI, therefore, requires a marked shift from the previous emphasis on a few historical systems embodying a number of famous methods, to a more generic approach - an approach which highlights those fundamental representation and search models that span all the different application areas and strategies of problem solving. Of course, since AI still does not have a well developed theory, references to seminal systems continues to fulfil an important role.

Both of the above books are good text books, characterised by a balanced coverage of Prolog and Lisp. They also reflect and consolidate much of the work of the past few years done in areas such as knowledge representation, machine learning, the work done under the heading of Expert Systems and even the recent work on neural networks. But the most important feature that they share is the accurate and up to date overall picture of the subject provided; the broad framework for the understanding of AI that is created without neglecting work of historical importance. There are still references to these works, but they are placed in perspective in relation to new developments.

The book of Luger & Stubblefield (L&S) is more language oriented than Firebaugh's book. A characteristic of L&S is that AI approaches to representation are related to the Object Oriented approach. Whereas L&S includes chapters on advanced AI programming techniques in Prolog and Lisp, it does not address pattern recognition, computer vision and robotics. (Firebaugh has chapters on each of these themes.) These omissions are understandable, since AI has diversified so much recently that it is difficult to cover all applications in one book.

If I had to select one of the books, it would be L&S. Although L&S gives poor coverage of Machine Learning, the book's overall presentation is very good. In particular, the chapters are well-organised, and the overall approach to AI - starting with the core aspects of *representation* and *search*, followed by chapters on AI languages - is coherent. The authors also make very good use of graphical representations and illustrations to convey ideas.

Books Received

The following books have been sent to SACJ. Anyone willing to review a book should contact the editor. The book will be sent to him for review, and may be kept provided that a review is received.

- D Bustard, J Elder & J Welsh, [1988], *Concurrent Program Structures*, Prentice-Hall Inc., Englewood Cliffs.
- R Cafolla & A D Kauffman, [1988], *Turbo Prolog Step by Step*, Merrill Publishing Company, Columbus, Ohio.
- S Hekmatpour, [1988], *Introduction to LISP and Symbol Manipulation*, Prentice-Hall Inc., Englewood Cliffs.
- K L Clark & F G McCabe, [1984], *micro-PROLOG: Programming in Logic*, Prentice-Hall Inc., Englewood Cliffs.
- D Crookes, [1988], *Introduction to Programming in Prolog*, Prentice-Hall Inc., Englewood Cliffs.
- M J C Gordon, [1988], *Programming Language Theory and its Implementation*, Prentice-Hall Inc., Englewood Cliffs.
- J G Hughes, [1988], *Database Technology : A software engineering approach*, Prentice-Hall Inc., Englewood Cliffs.
- R Milner, [1989], *Communication and Concurrency*, Prentice-Hall Inc., Englewood Cliffs.
- T J Myers, [1988], *Equations, Models and Programs*, Prentice-Hall, Inc., Englewood Cliffs.
- N C Rowe, [1988], *Artificial Intelligence through Prolog*, Prentice-Hall Inc., Englewood Cliffs.
- D A Protopapas, [1988], *Microcomputer Hardware Design*, Prentice-Hall Inc., Englewood Cliffs.
- H Eisner, [1988], *Computer-aided Systems Engineering*, Prentice-Hall Inc., Englewood Cliffs.
- S H Unger, [1989], *The essence of logic circuits*, Prentice-Hall Inc., Englewood Cliffs.
- R J Young, [1989], *Practical Prolog*, Van Nostrand Reinhold, New York.

How to access America's technical resources

You're engaged in expert systems, developments that are taking you close to the leading edge of technology.

You need specialized software, cards, accessories — products that are not being imported into South Africa.

We can get them for you!

At Sourcelink we have established on-line communication, by satellite, with our own purchasing organization in the United States.

We can get you a quotation on any software package or item of equipment you require within twenty four hours, and deliver it to your desk within fourteen to twenty-one days. Far quicker than by any other method. And at prices that are more than competitive.

And if you are not sure that the item you want exists, let us have your specification. For a modest fee we will carry out a search and tell you which product best fulfills your needs.

SOURCELINK

Your shopping service in the United States

(011) 728-1271/2

Ivylink, 103 Grant Avenue, Norwood

NOTES FOR CONTRIBUTORS

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as a Communications or Viewpoints. While English is the preferred language of the journal papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted **in triplicate** to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines.

- Use double-space typing on one side only of A4 paper, and provide wide margins.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be on separate sheets of A4 paper, and should be numbered and titled. Figures should be submitted as original line drawings, and not photocopies.
- Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin. Distinguish clearly between such cases as:
  - upper and lower case letters;
  - the letter O and zero;
  - the letter I and the number one; and
  - the letter K and kappa.
- References should be listed at the end of the text **in alphabetic order** of the (first) author's surname, and should be cited in the text in square brackets. References should thus take the following form:

  [1] E Ashcroft and Z Manna, [1972], The translation of 'GOTO' programs to 'WHILE' programs, *Proceedings of IFIP Congress 71*, North-Holland, Amsterdam, 250-255.

  [2] C Bohm and G Jacopini, [1966], Flow diagrams, Turing machines and languages with only two formation rules, *Comm. ACM*, **9**, 366-371.

  [3] S Ginsburg, [1966], *Mathematical theory of context free languages,* McGraw Hill, New York.

Manuscripts *accepted* for publication should comply with the above guidelines, and may provided in one of the following three formats:

- in a **typed form** (i.e. suitable for scanning);
- as an **ASCII file** on diskette; or
- in **camera-ready** format.

A page specification is available on request from the editor, for authors wishing to provide camera-ready copies.

Charges

A charge per final page, scaled to reflect scanning, typesetting and reproduction costs, will be levied on papers accepted for publication. The costs per final page are as follows:

Typed format:	R80-00
ASCII format:	R60-00
Camera-ready format :	R20-00

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers will be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

South African Computer Journal

Number 1, January 1990
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 1, Januarie 1990
ISSN 1015-7999

Contents